Comparison of several solution methods for the Euler equations in a finite volume code

Tamil Iniyan Manokaran
Matrikelnummer 4744573

Examiner: Prof. Dr.-Ing. habil. Cord-Christian Rossow
Deutsches Zentrum für Luft- und Raumfahrt (DLR)
Institut für Aerodynamik und Strömungstechnik

Supervisor: Dr. habil. Stefan Langer
Deutsches Zentrum für Luft- und Raumfahrt (DLR)
Institut für Aerodynamik und Strömungstechnik

Submission date: 20 September 2018

Decleration

I, Tamil Iniyan Manokaran born on 27.05.1994 hereby declare that the project work entitled Comparison of several solution methods for the Euler equations in a finite volume code submitted to the Deutsches Zentrum für Luft- und Raumfahrt (DLR), Braunschweig is a record of an original work done by me under the guidance of Dr. habil. Stefan Langer, Deutsches Zentrum für Luft- und Raumfahrt (DLR), Braunschweig and this project work is submitted in the partial fulfillment of the requirements for the award of the degree of Master of Computational Sciences in Engineering. The results embodied in this thesis have not been submitted to any other University or Institute for the award of any other degree.

Braunschweig, 20.09.2018

Abstract

I would like to start the thesis with an Einstein's quote, which admired most "Make things as simple as possible, but not simpler".

One of the most popular and simple methods to derive a non-linear equation to steady state is diagonal implicit Runge-Kutta method. This method leads us to get a final linear equation. Then we would like to solve that linear equation with any of the iterative solvers to get efficiently an approximate solution.

These solution methods can be often used as smoother in a nonlinear multigrid algorithm. The actual solution depends on the approximation of the derivative of the nonlinear equation and on the iterative method to approximately solve the linear equation.

Goal of the Studienarbeit is the implementation of a two dimensional Euler code which approximates solutions on unstructured meshes. Solving the Euler equation with the help of several solution methods such as explicit multi-stage Runge-Kutta schemes accelerated by local time stepping, implicit scheme based on a derivative corresponding to a discretization of compact stencil, LU-SGS scheme for the given meshes and finally we would like to compare the results. The two dimensional finite volume code, which implements the discretization of the Euler equations in two dimension is developed based on the knowledge acquired from the lecture "Algorithmen zur Lösung der Euler und Navier-Stokes Gleichungen".

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Iterative solution methods are often used to approximate solutions of nonlinear equations. For this to be done we need to linearize the nonlinear equation by approximating the derivative and then make use of any iterative method to approximately solve the linear equation. This approximate solution is used to update the state of the outer nonlinear iteration. PDE or integral equations are often discretized and solved approximately by the following way: Mesh generation, Governing Equation, Discretization, Algorithm implementation, comparison of results.

In the Mesh generation step meshes are generated. But in our thesis we are not going to generate the mesh, instead we are going to use the already generated mesh of NACA0012 airfoil. Read the available grid file, which means calculation of number of elements, faces, edges, etc. and calculation of normal, surface area, etc. The elements in the mesh are triangle and quadrilateral, that is a hybrid or unstructured mesh.

The Governing equation step deals with the integral form of the Euler equation in conservative form, that is the unknowns are represented as conservative variables. Then the next step is very important to address the main issues of the discretization of the integral equations. All the algorithms and their required functions for solving the Euler equation are discussed in Solution algorithm step. In our thesis the used solution algorithms are implicit scheme, LU-SGS scheme and explicit scheme. The Implicit and LU-SGS scheme are differentiated by approximating the derivative of the residual and the LU-SGS scheme is reduced to a single stage. The explicit scheme is the straight forward method. Then in the Numerical examples step we can be able to plot and compare our results with different test cases. Finally we can give the conclusion of the current work and small introduction about future work.

Chapter 2

Governing Equations

2.1 Governing equations in integral form

In this section we are going to represent the governing equations, which are required for this thesis and are referred from the lecture notes [1]. Let us consider the bounded domain $\Omega \subset \mathbb{R}^2$. The governing equations are derived from the fundamental laws of conservation of mass, momentum and energy. The first named as the *conservation of mass*.

$$\frac{d}{dt}\int_{\Omega} \rho(x,t)dx \; + \; \int_{\partial\Omega} \langle (\rho u)(x,t), n\rangle \; ds = 0. \tag{2.1a}$$

The second named as the *conservation of momentum*.

$$\frac{d}{dt}\int_{\Omega} (\rho u)_i(x,t) \; dx \; + \; \int_{\partial\Omega} \langle (\rho u)_i(x,t) \; u(x,t) \; + \; P(x,t)e_i, n\rangle \; ds = 0. \tag{2.1b}$$

The third named as the *conservation of Energy*.

$$\frac{d}{dt}\int_{\Omega} (\rho E)(x,t) \; dx \; + \; \int_{\partial\Omega} \langle \rho(x,t)H(x,t)u(x,t), n\rangle \; ds = 0. \tag{2.1c}$$

The quantities $\rho(x,t)$, $u(x,t) = (u_1(x,t), u_2(x,t))^T$, $E(x,t)$ and

$$H(x,t) := E(x,t) + P(x,t)/\rho(x,t) \tag{2.2}$$

are density, velocity, the specific total energy and the enthalpy of the fluid. n is the unit outward normal. P is the pressure defined by the state equation given as

$$P(x,t) := (\gamma - 1)\,\rho(x,t)\left(E(x,t) - \frac{\|u\|_2^2}{2}\right), \tag{2.3}$$

γ is the gas-dependent ratio of specific heat, which is given by 1.4 for air.

2.2 Integral form of euler equation

Let us consider the domain to describe the flow effects $\Omega \subset \mathbb{R}^2$, i.e., an open and connected set, and an integral $[0,T) \subset \mathbb{R}$, $T > 0$, the RANS equations in conservative form. The fundamental laws of conservation of mass, momentum and energy resulting a system of non-linear conservation laws. The governing equations can be expressed in the general form as

$$\frac{d}{dt}V_\Omega(W)(t) + R_{\partial\Omega}(W)(t) = 0, \qquad t \in [0,T) \tag{2.4a}$$

where the integral operators V_Ω and $R_{\partial\Omega}$ are given by

$$V_\Omega(W)(t) := \int_\Omega W(x,t)\, dx. \tag{2.4b}$$

For Euler equation,

$$R_{\partial\Omega}(W)(t) := \int_{\partial\Omega} \langle f_c(W), n\rangle\, ds, \tag{2.4c}$$

where $W(x,t)$ is the conservative variable,

$$W(x,t) = \begin{pmatrix} \rho(x,t) \\ \rho(x,t)u_1(x,t) \\ \rho(x,t)u_2(x,t) \\ \rho(x,t)E(x,t) \end{pmatrix},$$

The term $f_c(W)$ describes the convective contribution

$$f_c(W) := \begin{pmatrix} \rho u \\ (\rho u_1)u + Pe_1 \\ (\rho u_2)u + Pe_2 \\ \rho H u \end{pmatrix},$$

and $\langle f_c(W), n\rangle$ is called the convective flux in normal direction n, which is given by

$$\langle f_c(W), n\rangle = \begin{pmatrix} \rho(x,t)V \\ (\rho u_1)(x,t)V + P(x,t)n_1 \\ (\rho u_2)(x,t)V + P(x,t)n_2 \\ \rho(x,t)H(x,t)V \end{pmatrix},$$

where,

$$V := \langle u(x,t), n\rangle = u_1 n_1 + u_2 n_2 \tag{2.5}$$

describes the normal velocity. The speed of sound is denoted as a, dimensionless Mach number is denoted as M are defined by

$$a := \sqrt{\frac{\gamma P}{\rho}}, \qquad M := \frac{\|u\|_2^2}{a}. \tag{2.6}$$

The speed of sound a can be reformulated by using the equation of state as

$$a^2 = (\gamma - 1)\left(H - \frac{\|u\|_2^2}{2}\right).$$

Equation (2.4a) is rewritten as,

$$\frac{d}{dt}\int_\Omega W(x,t)\,dx + \int_{\partial\Omega} \langle f_c(W), n\rangle\,ds = 0, \tag{2.7}$$

which is the ***integral form of the Euler equation***.

2.3 Governing equations in differential form

The integral equation (2.7) hold for any infinitely small sub domain, here (2.4a) holds for any small domain and for the whole domain one can obtain the differential form, to reformulate the equations 2.1a, 2.1b, 2.1c into differential form, we apply Gauss-Divergence theorem.

Gauss-Divergence theorem is given as

$$\int_\Omega \operatorname{div} f_c(W)\,dx = \int_{\partial\Omega} \langle f_c(W), n\rangle\,ds. \tag{2.8}$$

The fundamental laws of conservation of mass, momentum and energy in differential form can be rewritten as,

$$\int_\Omega \frac{\partial \rho(x,t)}{\partial t} dx \;+\; \int_\Omega \operatorname{div}((\rho u)(x,t))dx \;=\; 0, \tag{2.9a}$$

$$\begin{aligned}\int_\Omega \frac{\partial}{\partial t}(\rho u)_i(x,t)dx \;+\; \int_\Omega [&\operatorname{div}((\rho u_i)(x,t)u(x,t)) \\ &+\; \operatorname{div}(P(x,t)e_i)]dx \;=\; 0,\end{aligned} \tag{2.9b}$$

$$\int_\Omega \frac{\partial}{\partial t}(\rho E)(x,t)dx \;+\; \int_\Omega \operatorname{div}(\rho(x,t)H(x,t)u(x,t))dx = \;0. \tag{2.9c}$$

Equation 2.9a - 2.9c hold for all bounded, open subset Ω, one obtains the differential equations

$$\frac{\partial}{\partial t}\rho(x,t) + \sum_{j=1}^{2}\frac{\partial}{\partial x_j}(\rho u)(x,t) = 0, \tag{2.10a}$$

$$\frac{\partial}{\partial t}(\rho u_i)(x,t) + \sum_{j=1}^{2}\left(\frac{\partial}{\partial x_j}(\rho u_i u_j)(x,t) + \delta_{ij}\frac{\partial}{\partial x_j}P(x,t)\right) = 0, \tag{2.10b}$$

$$\frac{\partial}{\partial t}(\rho E)(x,t) + \sum_{j=1}^{2}\frac{\partial}{\partial x_j}(\rho(x,t)H(x,t)u(x,t)) = 0. \tag{2.10c}$$

2.4 Differential form of euler equation

The differential form of the Euler equation, which is derived from the integral equation is given as

$$\frac{\partial}{\partial t}W(x,t) + \mathrm{div} f_c(W) = 0 \tag{2.11}$$

where $W(x,t)$ is the conservative variable, $f_c(W)$ is the convective term described in the Section 2.2.

Chapter 3

Discretization

3.1 Computational Mesh

This section deals about computation of the meshes, which are referred from the lecture notes [1]. Let us assume that the bounded computational domain is covered by a given finite set of domains $\{\Omega_i\}_{i=1,...N_{elem}}$.

Let $\Omega \subset \mathbb{R}^2$ be a bounded domain. Assume that there exists a finite set of open domains $\{\Omega_i\}_{i=1,...N_{elem}}$, $\Omega_i \subset \mathbb{R}^2$, $\Omega_i \neq \emptyset$, covering Ω, that is

$$\Omega_i \subset \Omega, \qquad \overline{\Omega} = \bigcup_{i=1}^{N_{elem}} \overline{\Omega_i}, \qquad \Omega_i \cap \Omega_j = \emptyset, \qquad i \neq j.$$

Then the set

$$M := \{\Omega : i = 1, ..., N_{elem}\}$$

is called a mesh or a grid or a decomposition covering Ω.

For two dimensional case, the decomposition M is called feasible if for all $i \neq j$ either $\overline{\Omega_i} \cap \overline{\Omega_j} = \emptyset$ or one of the following conditions hold:

a) $\overline{\Omega_i} \cap \overline{\Omega_j} \neq \emptyset$ and $\overline{\Omega_i}$ and $\overline{\Omega_j}$ share exactly one corner, or

b) $\overline{\Omega_i} \cap \overline{\Omega_j} \neq \emptyset$ and $\overline{\Omega_i}$ and $\overline{\Omega_j}$ share exactly one edge, or

A feasible decomposition M of $\Omega \subset \mathbb{R}^2$ is called a ***triangulation or a finite volume mesh***. It is noted that the considered meshes consist of triangles and quadrilaterals only.

3.1.1 Common Definitions

(a) The volume of Ω is denoted by

$$\text{vol}(\Omega) := \int_{\Omega} 1 \, dx.$$

(b) The point $x \in \mathbb{R}^2$ is called barycenter of Ω

$$x_i := \frac{1}{\text{vol}(\Omega)} \int_{\Omega} y_i \, dx, \qquad i = 1, 2.$$

(c) If edge(face) is defined as $\overline{\Omega_i} \cup (\mathbb{R}^2 \setminus \Omega) \neq 0$, then the boundary edge(face) is defined as

$$e_{i,bdry} := \overline{\Omega_i} \cup (\mathbb{R}^2 \setminus \Omega).$$

(d) The surface area of the face e_{ij} is denoted by

$$\text{svol}(e_{ij}) := \int_{\partial\Omega_i \cap \Omega_j} 1 \, ds.$$

(e) The surface area of the boundary face $e_{i,bdry}$ is denoted by

$$\text{svol}(e_{i,bdry}) := \int_{\partial\Omega_i \cap (\mathbb{R}^2 \setminus \Omega)} 1 \, ds.$$

(f) The outer unit normal vector of the face e_{ij} is denoted by

$$n_{e_{ij}} = n_{ij} = (n_{1,ij}, n_{2,ij}).$$

(g) The outer unit normal vector of the face $e_{i,bdry}$ is denoted by

$$n_{e_{i,bdry}} = n_{i,bdry} = (n_{1,i,bdry}, n_{2,i,bdry}).$$

3.1.2 Computation of area/volume of triangle

Let us consider the element Ω be a triangle with coordinates [P1, P2, P3] as shown in Fig. 3.1. From the available general definition in the section 3.1.1 the area/volume of the triangle is computed mathematically as follows

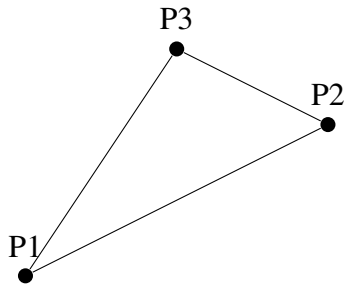


Figure 3.1: Area of triangle

$$\begin{aligned}\text{vol}(\Omega) =& \frac{1}{2}[(X_{P1}-X_{P2})(Y_{P1}+Y_{P1})+(X_{P2}-X_{P3})(Y_{P2}+Y_{P3}) \\ &+(X_{P3}-X_{P1})(Y_{P3}+Y_{P1})].\end{aligned}$$

3.1.3 Computation of area/volume of quadrilateral

Let us consider the element Ω be a quadrilateral with coordinates [P1, P2, P3, P4] as shown in Fig. 3.2. From the available general definition in the section 3.1.1 the area/volume of the quadrilateral is computed mathematically as follows

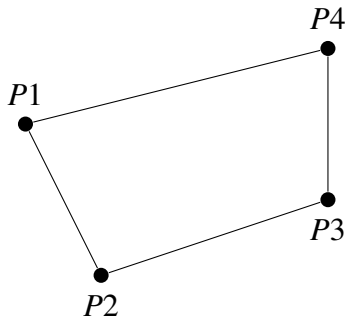


Figure 3.2: Area of quadrilateral

$$\text{vol}(\Omega) = \frac{1}{2}[(X_{P4}-X_{P2})(Y_{P1}+Y_{P3})+(X_{P1}-X_{P3})(Y_{P2}+Y_{P4})].$$

3.2 Discretization

As we mentioned in the introduction part, this is one of the important steps for this thesis referred from [1]. So let us consider the differential form of the Euler

equation,

$$\frac{\partial}{\partial t}W(x,t) + \operatorname{div} f_c(W) = 0, \tag{3.1}$$

multiplying the suitable test function v, apply the Gauss Divergence theorem and identity, which is given below to the equation (3.1)

$$\operatorname{div}(v f_c(W)) = v \operatorname{div}(f_c(W)) + \langle f_c(W), \operatorname{grad} v \rangle,$$

we will get the weak form of Euler equation as

$$\int_\Omega v \frac{dW}{dt}\, dx - \int_\Omega \langle \operatorname{grad} v, f_c(W) \rangle\, dx + \int_{\partial\Omega} \langle v f_c(W), n \rangle\, ds = 0. \tag{3.2}$$

Equation (3.2) holds for all possible test functions v. In this thesis we are going to do finite volume discretization, in which we approximate the unknown function W representing it by a sum of constant ansatz functions. For this purpose we are going to define the indicator function, which is described as follows

$$\mathbb{1}_{\Omega_i}(x) := \begin{cases} 1, & x \in \Omega_i \\ 0, & else \end{cases}$$

and W is approximated by using the constant function W_h,

$$W(x,t) \approx W_h(x,t) \approx \sum_{i=1}^{N_{elem}} W_i(t)\, \mathbb{1}_{\Omega_i}(x),$$

it is always the case that,

$$\operatorname{grad}(W_i \mathbb{1}_{\Omega_i})|_{\Omega_i} = 0, \quad i = 1, ..., N_{elem}$$

so, now the only test function is

$$v(x) = \sum_{i=1}^{N_{elem}} \mathbb{1}_{\Omega_i}(x), \quad i = 1, ..., N_{elem}$$

As we know our test function is a constant function, which gives $\operatorname{grad} v(x) = 0$. And we substitute all those in our weak form of Euler equation (3.2), then we arrive with the discretized form of Euler equation with finite volume discretization, which is given as follows

$$\sum_{i=1}^{N_{elem}} \left(\operatorname{vol}(\Omega_i) \frac{dW_i}{dt} + \int_{\partial\Omega_i} \langle f_c(W_h), n \rangle\, ds \right) = 0. \tag{3.3}$$

To do the discretization we need the quadrature formula for the volume and surface integrals. But the surface integrals lead us into the trouble when the vector W

of the conserved variable is discontinuous across the faces. These problems are named as Riemann problems and to address these issues the concept of a numerical flux function has been introduced, which is the stable numerical method in the up-winding scheme. These flux functions are also often called as Riemann solver. Hence the surface integral in equation(3.3) is written as follows,

$$\int_{\partial\Omega_i} \langle f_c(W_h), n\rangle \, ds = \sum_{j\in N(i)} \int_{e_{ij}} H(W_L(i,N(i)), W_R(j,N(j)), n) \, ds, \tag{3.4}$$

where H is the numerical flux function and the neighbors of the vertex i and j are denoted by $N(i)$ & $N(j)$ respectively. Here, $W_L(i,N(i))$ and $W_R(j,N(j))$ are the functions, which is used to compute the flux over the face e_{ij} are not only based directly on the W_i, W_j, but also on the neighbor stencil. In detail the states are reconstructed as follows,

$$W_L(i,N(i)) = W(W_i, W_{k,k\in N(i)}), \qquad W_R(j,N(j)) = W(W_j, W_{k,k\in N(j)})$$

As stated before the state W_L may depend on the coefficient vector W_i and all the surrounding coefficient vectors W_k, $k \in N(i)$, and similarly the state W_R may depend on the coefficient vector W_j and all the surrounding coefficient vectors W_k, $k \in N(j)$. Hence the equation with the approximation of surface integral is given as,

$$\int_{e_{ij}} \langle f_c(W_h), n\rangle \, ds \approx \text{svol}(e_{ij}) \, H(W_L(i,N(i)), W_R(j,N(j)), n_{e_{ij}}),$$

where surface area of the face e_{ij} is already defined in Section 3.1.4. In our case we are considering only the fluxes corresponding to the direct neighbors. So the modified equation is given as,

$$\int_{e_{ij}} \langle f_c(W_h), n\rangle \, ds \approx \text{svol}(e_{ij}) \, H(W_i, W_j, n_{e_{ij}}).$$

The numerical flux function in normal direction n by corresponding to a first order Roe scheme is defined as

$$H^{1st,Roe}(W_i, W_j, n) := \frac{1}{2} \left[\langle f_c(W_i), n\rangle + \langle f_c(W_j), n\rangle\right] - \frac{1}{2} |A_{ij}^{Roe}| \, (W_j - W_i). \tag{3.5}$$

The operator A_{ij}^{Roe} is given by

$$A_{ij}^{Roe} := \frac{\partial \langle f_c(W_{ij,Roe}), n\rangle}{\partial W},$$

here 'Roe' refers that we use Roe averaged variables to evaluate the corresponding term on the edge (face) ij,

$$\rho_{ij,Roe} := \sqrt{\rho_i \rho_j}, \tag{3.6a}$$

$$(u_{ij,Roe})_k := \frac{(u_i)_k\sqrt{\rho_i} + (u_j)_k\sqrt{\rho_j}}{\sqrt{\rho_i} + \sqrt{\rho_j}}, \qquad k = 1,2 \tag{3.6b}$$

$$H_{ij,Roe} := \frac{H_i\sqrt{\rho_i} + H_j\sqrt{\rho_j}}{\sqrt{\rho_i} + \sqrt{\rho_j}}, \tag{3.6c}$$

$$a^2_{ij,Roe} := (\gamma - 1)\left(H_{ij,Roe} - \frac{1}{2}\left\|u_{ij,Roe}\right\|_2^2\right), \tag{3.6d}$$

$$E_{ij,Roe} := H_{ij,Roe} - \frac{a^2_{ij,Roe}}{\gamma}. \tag{3.6e}$$

Equation (3.6d) represents square of the corresponding Roe averaged speed of sound and equation (3.6e) represents the total energy.

3.3 Discretization of boundary conditions

All boundary conditions considered here are implemented using a flux formulation [2]. The used boundary condition in our thesis are given as,

1. Farfield boundary condition
2. Slip wall boundary condition

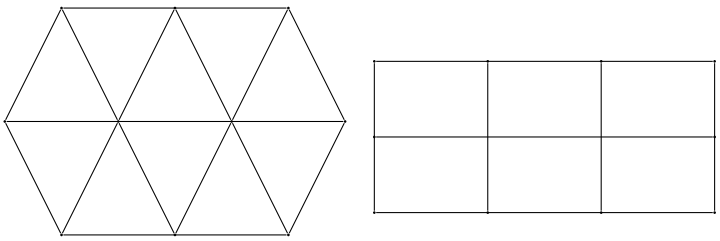


Figure 3.3: Examples of primary mesh

The flux over the edge $e_{i,bdry}$ needs to satisfy the corresponding boundary condition using flux formulations. Here the coefficients W_i of the ansatz function corresponding to the element Ω_i are given. In order to compute the flux we need to define an an outer artificial state, which is given as

$$W_{i,bdry} = \left(\rho_{i,bdry}(t), (\rho u)_{i,bdry}(t), (\rho E)_{i,bdry}(t)\right)^T.$$

This outer artificial state should satisfy the boundary condition for the boundary edge $e_{i,bdry}$ and the approximated boundary integral can be given by,

$$\begin{aligned}\int_{\partial\Omega_i} \langle f_c(W),\, n\rangle \, ds &\approx \sum_{i=1}^{N_{bdry}} \int_{e_{i,bdry}} H^{1st,Roe}\,(W_i, W_{i,bdry}, n_{i,bdry})\, ds \\ &\approx \sum_{i=1}^{N_{bdry}} \text{svol}\,(e_{i,bdry})\, H^{1st,Roe}\,(W_i, W_{i,bdry}, n_{i,bdry}).\end{aligned}$$

3.3.1 Farfield boundary condition

Using this boundary condition we can find the inflow of free-stream conditions of the fluid. For the given angle of attack α, the outer state can be found by

$$W_{i,bdry} := W_\infty := (\rho_\infty,\, \cos\alpha\rho_\infty u_\infty,\, \sin\alpha\rho_\infty u_\infty,\, \rho_\infty E_\infty)^T.$$

3.3.2 Slip wall boundary condition

The slip wall boundary conditions are used for the inviscid flow problem. To ensure a vanishing normal velocity the outer state can be found by

$$\begin{aligned}\rho_{i,bdry} &:= \rho_i, \\ (\rho u)_{i,bdry} &:= (\rho u)_i - 2\langle(\rho u)_i, n_{i,bdry}\rangle n_{i,bdry}, \\ \rho_{i,bdry}E_{i,bdry} &:= \rho_i E_i.\end{aligned}$$

3.4 Derivative of the convective flux

In this part we are going to do the implementation of the Matrix Dissipation, which is referred from the paper "*Investigation and application of point implicit Runge-Kutta methods to inviscid flow problems*" [3]. Here the determination of the eigenvalues and eigenvectors does not require rewriting the derivative in primitive variables and convert it back. With this there is an advantage of adding the turbulent parts if it is necessary in future. But in our thesis we are not looking

for any turbulent parts and formulation of Matrix Dissipation can be implemented straightforward.

Let us consider the convective part of equation (2.4), mentioned in the page 10,

$$\langle f_c(W), n\rangle = V\begin{pmatrix}\rho\\ \rho u_1\\ \rho u_2\\ \rho E\end{pmatrix} + P\begin{pmatrix}0\\ n_1\\ n_2\\ 0\end{pmatrix} + VP\begin{pmatrix}0\\ 0\\ 0\\ 1\end{pmatrix} = VW + P\begin{pmatrix}0\\ n_1\\ n_2\\ V\end{pmatrix}.$$

Now the derivative of $\langle f_c(W), n\rangle$ is given by

$$\frac{\partial\langle f_c(W), n\rangle}{\partial W} = VI + W\frac{\partial V(W)}{\partial W} + \begin{pmatrix}0\\ n_1\\ n_2\\ V\end{pmatrix}\frac{\partial P(W)}{\partial W} + P\begin{pmatrix}0\\ 0\\ 0\\ \dfrac{\partial V(W)}{\partial W}\end{pmatrix}. \tag{3.7}$$

The derivative of the normal velocity, $V = \langle u, n\rangle$ is given by

$$\frac{\partial V(W)}{\partial W} = \frac{1}{\rho}\begin{pmatrix}-V\\ n_1\\ n_2\\ 0\end{pmatrix}^T. \tag{3.8}$$

Apply equation 3.8 in 3.7

$$\frac{\partial\langle f_c(W), n\rangle}{\partial W} = VI + \frac{1}{\rho}\begin{pmatrix}\rho\\ \rho u_1\\ \rho u_2\\ \rho E\end{pmatrix}(-V, n_1, n_2, 0) + \begin{pmatrix}0\\ 0\\ 0\\ \dfrac{p}{\rho}\end{pmatrix}(-V, n_1, n_2, 0) + \begin{pmatrix}0\\ n_1\\ n_2\\ V\end{pmatrix}\frac{\partial P(W)}{\partial W} \tag{3.9a}$$

$$= VI + \begin{pmatrix}1\\ u_1\\ u_2\\ H\end{pmatrix}(-V, n_1, n_2, 0) + \begin{pmatrix}0\\ n_1\\ n_2\\ V\end{pmatrix}\frac{\partial P(W)}{\partial W}. \tag{3.9b}$$

Now we need derivation of pressure P, which is given by

$$\frac{\partial P(W)}{\partial W} = (\gamma - 1)\left(\frac{\|u\|_2^2}{2}, -u_1, -u_2, 1\right). \tag{3.10}$$

Let us write the full matrix of derivative of $f_c \cdot n$ by applying equation 3.10 in 3.9b

$$\frac{\partial \langle f_c(W), n\rangle}{\partial W} = \begin{bmatrix} V & 0 & 0 & 0 \\ 0 & V & 0 & 0 \\ 0 & 0 & V & 0 \\ 0 & 0 & 0 & V \end{bmatrix} + \begin{bmatrix} -V & n_1 & n_2 & 0 \\ -Vu_1 & u_1 n_1 & u_1 n_2 & 0 \\ -Vu_2 & u_2 n_1 & u_2 n_2 & 0 \\ -VH & Hn_1 & Hn_2 & 0 \end{bmatrix}$$
$$+ \begin{bmatrix} 0 & 0 & 0 & 0 \\ \frac{\gamma-1}{2}\|u\|_2^2 n_1 & -(\gamma-1)u_1 n_1 & -(\gamma-1)u_2 n_1 & (\gamma-1)n_1 \\ \frac{\gamma-1}{2}\|u\|_2^2 n_2 & -(\gamma-1)u_1 n_2 & -(\gamma-1)u_2 n_2 & (\gamma-1)n_2 \\ \frac{\gamma-1}{2}\|u\|_2^2 V & -(\gamma-1)u_1 V & -(\gamma-1)u_2 V & (\gamma-1)V \end{bmatrix}$$

$$\frac{\partial \langle f_c(W), n\rangle}{\partial W} = \begin{bmatrix} 0 & n_1 & n_2 & 0 \\ \frac{n_1\zeta_2\|u\|_2^2}{2} - u_1 V & n_1\zeta_3 u_1 + V & n_2 u_1 - n_1\zeta_2 u_2 & n_1\zeta_2 \\ \frac{n_2\zeta_2\|u\|_2^2}{2} - u_2 V & n_1 u_2 - n_2\zeta_2 u_1 & n_2\zeta_3 u_2 + V & n_2\zeta_2 \\ (\zeta_2\|u\|_2^2 - \gamma E)V & n_1\zeta_1 - \zeta_2 u_1 V & n_2\zeta_1 - \zeta_2 u_2 V & \gamma V \end{bmatrix} \tag{3.11}$$

where

$$\zeta_1 = \gamma E - \Phi, \quad \zeta_2 = \gamma - 1, \quad \zeta_3 = 2 - \gamma, \quad \Phi = \frac{1}{2}(\gamma-1)\|u\|_2^2$$

3.5 Eigenvalues and Eigenvectors of the derivative of the convective flux

We define the vectors

$$a_1 := (1, u_1, u_2, H)^T$$
$$a_2 := (0, n_1, n_2, V)^T$$
$$b_1 := (-V, n_1, n_2, 0)^T$$
$$b_2 := \left(\frac{\partial P(W)}{\partial W}\right)^T$$

Then using Equation 3.9b the derivative of the convective flux can be assembled as

$$\frac{\partial \langle f_c(W), n\rangle}{\partial W} = VI + a_1 b_1^T + a_2 b_2^T. \tag{3.12}$$

Then we notice that $x \in \mathbb{R}^4$ is eigenvector with eigenvalue V of $\frac{\partial(f_c.n)(W)}{\partial W}$ if and only if the orthogonality relations

$$b_1^T x = b_2^T x = 0 \tag{3.13}$$

hold. It can be easily verified that the vectors

$$y_1 = \begin{pmatrix} 1 \\ u_1 \\ u_2 \\ \|u\|_2^2 \end{pmatrix}, \qquad y_2 = \begin{pmatrix} 0 \\ n_2 \\ -n_1 \\ n_2u_1 - n_1u_2 \end{pmatrix}, \qquad y_3 = \begin{pmatrix} 0 \\ -n_2 \\ n_1 \\ n_1u_2 - n_2u_1 \end{pmatrix},$$

satisfy equation 3.13. However the vectors y_2, y_3 are linearly dependent since

$$y_2 + y_3 = 0.$$

Two linear-independent eigenvectors may be obtained as

$$g_1 := Ay_1 + ay_2$$
$$g_2 := Ay_1 + ay_3$$

where $a := \sqrt{\frac{\gamma P}{\rho}}$ denotes the speed of sound and $A := \sqrt{n_1^2 + n_2^2}$ denotes the surface area. In our case we have used the normal, which is normalized by surface area, so the value of surface area is equal to 1. To identify the remaining eigenvectors we use the relations

$$b_1^T a_1 = b_2^T a_2 = 0, \quad b_1^T a_2 = A^2 \quad \textit{and} \quad b_2^T a_1 = a^2, \tag{3.14}$$

$$b_1^T a_1 = (-V, n_1, n_2, 0) \begin{pmatrix} 1 \\ u_1 \\ u_2 \\ H \end{pmatrix} = -V + u_1n_1 + u_2n_2 = 0,$$

where, V is defined in equation (2.5) . similarly $b_2^T a_2 = 0$.

$$b_1^T a_2 = (-V, n_1, n_2, 0) \begin{pmatrix} 1 \\ n_1 \\ n_2 \\ V \end{pmatrix} = n_1^2 + n_2^2 = A^2,$$

$$b_2^T a_1 = (\gamma-1)\left(\frac{\|u\|_2^2}{2}, -u_1, -u_2, 1\right)\begin{pmatrix}1\\u_1\\u_2\\H\end{pmatrix} = \frac{\gamma P}{\rho} = a^2.$$

Then we compute using Equation(3.12)

$$\begin{aligned}\frac{\partial\langle f_c(W),n\rangle}{\partial W}(Aa_1+aa_2) &= VI(Aa_1+aa_2)+a_1b_1^T(Aa_1+aa_2)+a_2b_2^T(Aa_1+aa_2)\\ &= VAa_1+Vaa_2+Aa_1b_1^Ta_1+aa_1b_1^Ta_2+Aa_2b_2^Ta_1+aa_2b_2^Ta_2\\ &= VAa_1+Vaa_2+0+aa_1A^2+Aa_2a^2+0\\ &= VAa_1+Vaa_2+aa_1A^2+Aa_2a^2\\ &= (V+aA)(Aa_1+aa_2)\end{aligned}$$

similarly,

$$\begin{aligned}\frac{\partial\langle f_c(W),n\rangle}{\partial W}(Aa_1-aa_2) &= VAa_1+a^2Aa_2-Vaa_2-A^2aa_1\\ &= (V-aA)(Aa_1-aa_2)\end{aligned}$$

Hence we found the two additional eigenvalues $V+aA$ and $V-aA$ with corresponding eigenvectors $g_3 := Aa_1+aa_2$ and $g_4 := Aa_1-aa_2$.

3.6 Eigendecomposition of the derivative of the convective flux

The eigendecomposition of the derivative of the convective flux is given as

$$\frac{\partial\langle f_c(W),n\rangle}{\partial W} = G\,\Lambda\,G^{-1} \tag{3.15}$$

Where Λ is a diagonal matrix, which has on its diagonal the eigenvalues of $\frac{\partial \langle f_c(W), n \rangle}{\partial W}$ and columns of matrix G are the eigenvectors of $\frac{\partial \langle f_c(W), n \rangle}{\partial W}$

$$G = (g_1, g_2, g_3, g_4)$$

$$\Lambda = \begin{pmatrix} V \\ V \\ V + aA \\ V - aA \end{pmatrix}$$

We define the vectors

$$P_1 := \frac{\gamma - 1}{a^2} \begin{pmatrix} H - \|u\|_2^2 \\ u_1 \\ u_2 \\ -1 \end{pmatrix} \tag{3.16a}$$

$$P_2 := \begin{pmatrix} u_1 n_2 - u_2 n_1 \\ -n_2 \\ n_1 \\ 0 \end{pmatrix} \tag{3.16b}$$

$$P_3 := \begin{pmatrix} u_2 n_1 - u_1 n_2 \\ n_2 \\ -n_1 \\ 0 \end{pmatrix} \tag{3.16c}$$

and

$$q_1 := \frac{1}{2A^2} \left(A p_1^T - \frac{1}{a} P_2^T \right) \tag{3.17a}$$

$$q_2 := \frac{1}{2A^2} \left(A p_1^T - \frac{1}{a} P_3^T \right) \tag{3.17b}$$

$$q_3 := \frac{1}{2a^2 A} \left(b_2^T + \frac{a}{A} b_1^T \right) \tag{3.17c}$$

$$q_4 := \frac{1}{2a^2A}\left(b_2^T - \frac{a}{A}b_1^T\right) \tag{3.17d}$$

Then the inverse of G is given by

$$J := \begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{pmatrix} \qquad i.e.\ G^{-1} = J$$

Here we are going to use some of the important equations

$$P_1^T y_1 = \frac{\gamma-1}{a^2}\left(H - \|u\|_2^2 + \|u\|_2^2 - \frac{\|u\|_2^2}{2}\right) = \frac{\gamma-1}{a^2}\left(H - \|u\|_2^2\right) = 1,$$
$$P_1^T y_2 = P_1^T y_3 = P_2^T y_1 = P_3^T y_1 = 0,$$
$$P_2^T y_2 = -n_2^2 - n_1^2, \quad P_3^T y_3 = -n_2^2 - n_1^2,$$
$$P_2^T y_3 = n_2^2 + n_1^2, \quad P_3^T y_2 = n_2^2 + n_1^2,$$
$$b_1^T y_1 = b_1^T y_2 = b_1^T y_3 = 0,$$
$$b_2^T y_1 = b_2^T y_2 = b_2^T y_3 = 0,$$

and equation (3.14) to prove the assertion

$$q_1 g_1 = \frac{1}{2A^2}\left(A^2 P_1^T y_1 - \frac{A}{a}P_2^T y_1 + aAP_1^T y_2 - P_2^T y_2\right) = \frac{1}{2A^2}(A^2 + n_1^2 + n_2^2) = 1,$$
$$q_1 g_2 = q_1 g_3 = q_1 g_4 = 0,$$
$$q_2 g_2 = \frac{1}{2A^2}\left(A^2 P_1^T y_1 - \frac{A}{a}P_3^T y_1 + aAP_1^T y_3 - P_3^T y_3\right) = \frac{1}{2A^2}(A^2 + n_1^2 + n_2^2) = 1,$$
$$q_2 g_1 = q_2 g_3 = q_2 g_4 = 0,$$
$$q_3 g_3 = \frac{1}{2a^2A}\left(Ab_2^T a_1 + ab_1^T a_1 + ab_2^T a_2 + \frac{a^2}{A}b_1^T a_2\right) = \frac{1}{2a^2A}(a^2A + a^2A) = 1,$$
$$q_3 g_1 = q_3 g_2 = q_3 g_4 = 0,$$
$$q_4 g_4 = \frac{1}{2a^2A}\left(Ab_2^T a_1 - ab_1^T a_1 - ab_2^T a_2 + \frac{a^2}{A}b_1^T a_2\right) = \frac{1}{2a^2A}(a^2A + a^2A) = 1,$$
$$q_4 g_1 = q_4 g_2 = q_4 g_3 = 0.$$

3.7 Entropy fix

For a construction of the stable scheme, it is important to use the so-called entropy fix. Entropy fix is to avoid the instabilities when one of the eigenvalues is ≈ 0.

Let $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ be the eigenvalues, then the entropy fix is described as follows,

$$\begin{aligned}
\lambda_1 &:= \max\{|V|, \delta(|V| + aA)\}, \\
\lambda_2 &:= \max\{|V|, \delta(|V| + aA)\}, \\
\lambda_3 &:= \max\{|V + aA|, \delta(|V| + aA)\}, \\
\lambda_4 &:= \max\{|V - aA|, \delta(|V| + aA)\},
\end{aligned}$$

where δ is the user defined value. In our case we are using $\delta = 0.2$. Note that if the choice of $\delta = 0$, then no entropy fix is used.

Finally, we obtain equation (3.3) as given below

$$\frac{dW}{dt} = -M^{-1} R(W), \qquad \text{where} \quad M := \text{diag}(\text{vol}(\Omega_i)_{i=1,..,N_{elem}}). \tag{3.18}$$

We are only interested in approximating a steady state solution of equation (3.18). Hence the above equation is simplified as

$$-M^{-1} R(W) = 0 \tag{3.19}$$

Chapter 4

Solution Algorithms

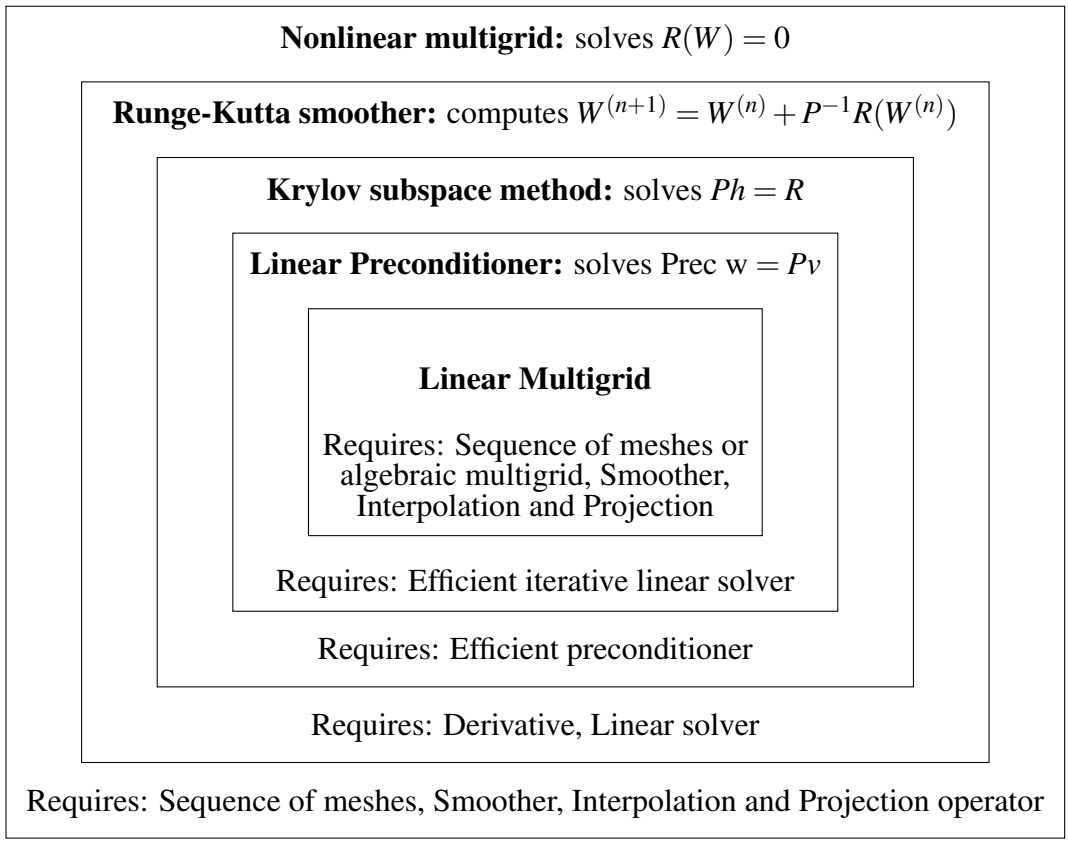


Figure 4.1: Algorithmical structure of nonlinear solution method

A general graphical overview to construct a powerful algorithm to solve a nonlinear equation is given in Figure 4.1, which is referred from [1]. It shows the connection of several required inputs. In our thesis the nonlinear algebraic system

of equations of interest is given by equation (3.19).

We can apply nonlinear multi grid method using an implicit Runge-Kutta smoother to solve the algebraic system of equations approximately and efficiently. Then we will get linear system of equations, they are solved approximately by using one of the iterative solution methods. We have to make use of the preconditioner to approximate efficiently a solution of the linear systems. To approximately solve efficiently the linear system linear multi grid methods can be applied.

4.1 Point implicit Runge-Kutta method

To solve the discretized flow equation (3.19) we consider multi stage diagonally implicit Runge-Kutta method given by the Butcher scheme (Table 4.1). Referred from the Journal "*Aglomeration multigrid methods with implicit Runge-Kutta smoothers applied to aerodynamic simulations on unstructured grids*" [4].

c	A
	b^T

Table 4.1: Butcher Scheme

where,

$$A := \begin{pmatrix} \alpha_{11} & 0 & \dots & 0 \\ \alpha_{21} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & \alpha_{s,s-1} & \alpha_{ss} \end{pmatrix}, \qquad b := \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \alpha_{s+1,s} \end{pmatrix} \qquad \text{and} \qquad c := \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \tag{4.1}$$

The stages of the implicit Runge-Kutta scheme and discrete evolution are given by

$$\begin{aligned} k_1(W) &= -M^{-1}R(W^{T_n} + \alpha_{11}\Delta t k_1(W)), \\ k_2(W) &= -M^{-1}R(W^{T_n} + \alpha_{21}\Delta t k_1(W) + \alpha_{22}\Delta t k_2(W)), \\ &\vdots \\ k_s(W) &= -M^{-1}R(W^{T_n} + \alpha_{s,s-1}\Delta t k_{s-1}(W) + \alpha_{ss}\Delta t k_s(W)), \\ W^{T_{n+1}} &= W^{T_n} + \alpha_{s+1,s}\Delta t k_s(W). \end{aligned} \tag{4.2}$$

To approximate a solution of nonlinear systems $k_1, \ldots, k_s$ we use one iteration of the Newton's method to approximate the root of the function

$$g_j(k_j(W)) := k_j(W) + M^{-1}R(W^{T_n} + \alpha_{j,j-1}\Delta t k_{j-1}(W) + \alpha_{jj}\Delta t k_j(W)).$$

And its derivative is given by

$$\frac{\partial g_j(k_j(W))}{\partial k_j(W)}[k_j(W)] = \mathrm{I} + \alpha_{jj}\Delta t M^{-1}\frac{\partial R}{\partial W}(W^{T_n} + \alpha_{j,j-1}\Delta t k_{j-1}(W) + \alpha_{jj}\Delta t k_j(W)),$$

and with an assumed initial guess $k_j^{(0)}(W) = 0$ the approximate root is given by

$$k_j(W) = -[P_j(W)]^{-1}(g_j(k_j^{(0)}(W))), \tag{4.3}$$

where

$$P_j(W) = \mathrm{I} + \alpha_{jj}\Delta t M^{-1}\frac{\partial R}{\partial W}[W^{T_n} + \alpha_{j,j-1}\Delta t k_{j-1}(W)].$$

Using these formulas the implicit Runge-Kutta method (4.2), where the inner Newton iteration is truncated after one step, can be represented by the algorithm

$$\begin{aligned}
k_1(W) &= -[P_1(W)]^{-1}M^{-1}R(W^{T_n}),\\
k_2(W) &= -[P_2(W)]^{-1}M^{-1}R(W^{T_n} + \alpha_{21}\Delta t k_1),\\
&\vdots\\
k_s(W) &= -[P_s(W)]^{-1}M^{-1}R(W^{T_n} + \alpha_{s,s-1}\Delta t k_{s-1}),\\
W^{T_{n+1}} &= W^{T_n} + \alpha_{s+1,s}\Delta t k_s(W).
\end{aligned}$$

Substitute $W^{(0)} := W^{T_n}$ and

$$W^{(j)} := W^{T_n} - \alpha_{j+1,j}\Delta t[P_j(W)]^{-1}M^{-1}R(W^{(j-1)})$$

Now the implicit Runge-Kutta method is reformulated as

$$\begin{aligned}
W^{(0)} &:= W^{T_n},\\
W^{(1)} &:= W^{(0)} - \alpha_{21}\Delta t P_1(W)^{-1}M^{-1}R(W^{(0)}),\\
&\vdots\\
W^{(s)} &:= W^{(0)} - \alpha_{s+1,s}\Delta t P_s(W)^{-1}M^{-1}R(W^{(s-1)}),\\
W^{T_n} &:= W^{(s)}.
\end{aligned} \tag{4.4}$$

Algorithm (4.4) indicates that for each stage the linear equation

$$P_j(W)h_j = \alpha_{j+1,j}\Delta t M^{-1}R(W^{(j-1)})$$

needs to be solved. This can be equivalently formulated by

$$\left((\Delta t)^{(-1)}M + \alpha_{jj}\frac{\partial R}{\partial W}\right)h_j = \alpha_{j+1,j}R(W^{(j-1)}). \tag{4.5}$$

4.2 Construction of preconditioner

As described in the section 3.2, an appropriate first order discretization is used to approximate the the exact derivative $\frac{\partial R}{\partial W}$. We are here neglecting the second order terms and assuming A_{ij}^{Roe} as locally constant, the derivative of the convective flux $\int_{\partial\Omega_i} f_c.n\ ds$ is formulated as,

$$\frac{\partial R_i}{\partial W_k} = \frac{1}{2}\begin{cases} \sum_{j\in N(i)} |A_{ij}^{Roe}|, & k = i, \\ -|A_{ij}^{Roe}|, & k \in N(i), \\ 0, & k \neq i,\ k \notin N(i). \end{cases} \tag{4.6}$$

We can simplify it further based on approximating terms of the first-order Jacobian by their spectral radius [5]. Equation (3.5) is replaced by a first-order scalar dissipative term,

$$\int_{\partial\Omega_i} f_c.n\ ds \approx \sum_{j\in N(i)} \frac{1}{2}\left[\langle f_c(W_i), n\rangle\ +\ \langle f_c(W_j), n\rangle\right]\ -\ \frac{1}{2}\rho(A_{ij}^{Roe})\,(W_j - W_i) \tag{4.7}$$

and $\rho(A_{ij}^{Roe})$ denotes the spectral radius of A_{ij}^{Roe}. Corresponding to Equation (4.6), the derivative of the convective flux is $\int_{\partial\Omega_i} f_c.n\ ds$ approximated by

$$\frac{\partial R_i^{\text{scalar}}}{\partial W_k} = \frac{1}{2}\begin{cases} \sum_{j\in N(i)} \rho(A_{ij}^{Roe}), & k = i, \\ -\rho(A_{ij}^{Roe}), & k \in N(i), \\ 0, & k \neq i,\ k \notin N(i). \end{cases} \tag{4.8}$$

Then for steady state computations the time step Δt in equation 4.5 is replaced by $\Delta T := \text{diag}(\text{diag}(\Delta t_i))$, where

$$\Delta t_i := \text{CFL}.\text{vol}(\Omega_i)\left[\sum_{j\in N(i)} \frac{1}{2}(|V_{ij}| + a_{ij}\text{svol}(e_{ij}))\right]^{-1} \tag{4.9}$$

Finally equation 4.5 is replaced by

$$\left((\Delta T)^{(-1)}M + \alpha_{jj}\frac{\partial R}{\partial W}\right)h_j = \alpha_{j+1,j}R(W^{(j-1)}) \tag{4.10}$$

From the above equation the final preconditioner is given as

$$\text{Prec}_j := (\Delta T)^{-1}M + \alpha_{jj}\frac{\partial R}{\partial W}(W^{(j-1)}) \tag{4.11}$$

$$\text{Prec}_j^{\text{scalar}} := (\Delta T)^{-1}M + \alpha_{jj}\frac{\partial R^{\text{scalar}}}{\partial W}(W^{(j-1)}) \tag{4.12}$$

4.3 Iterative solution methods for linear equations

In order to approximate efficient solution of the above linear equation (4.10) we are going to use any of the iterative solution methods like (Block) Jacobi or (Block) Gauss-Seidel [4]. Let P_{ij} denotes the entries of the preconditioner, then Block Jacobi method and Block Gauss-Seidel method are given as follows,

$$h_i^{k+1} = (P_{i,i})^{-1}\left(b_i - \sum_{j=1, j\neq i}^{N_{elem}} P_{i,j} h_j^{(k)}\right) \quad \text{for } i = 1,\ldots,N_{elem}, \tag{4.13}$$

$$h_i^{k+1} = (P_{i,i})^{-1}\left(b_i - \sum_{j=1}^{i-1} P_{i,j} h_j^{(k+1)} - \sum_{j=i+1}^{N_{elem}} P_{i,j} h_j^{(k)}\right) \quad \text{for } i = 1,\ldots,N_{elem}. \tag{4.14}$$

It is noted that in our code we are realizing our $(P_{i,i})^{-1}$ by using the pivoted LU decomposition.

4.4 Efficient smoother

In order to obtain low-cost efficient smoother our algorithm(4.6) reduces to the approximate solution of

$$\text{Prec}_j h_j = \alpha_{j+1,j}\, R(W^{(j-1)}), \tag{4.15}$$

and as a consequence Algorithm simplifies to

$$\begin{aligned} W^{(0)} &:= W^n, \\ W^{(1)} &:= W^{(0)} - \alpha_{21}\text{Prec}_1^{-1} R(W^{(0)}), \\ &\vdots \\ W^{(s)} &:= W^{(0)} - \alpha_{s+1,s}\text{Prec}_s^{-1} R(W^{(s-1)}), \\ W^{n+1} &:= W^{(s)}. \end{aligned} \tag{4.16}$$

It can also be the case that we can use the freezing preconditioner **Prec** on the first stage, that is

$$\begin{aligned} W^{(0)} &:= W^n, \\ W^{(1)} &:= W^{(0)} - \alpha_{21}\text{Prec}_1^{-1} R(W^{(0)}), \\ &\vdots \\ W^{(s)} &:= W^{(0)} - \alpha_{s+1,s}\text{Prec}_1^{-1} R(W^{(s-1)}), \\ W^{n+1} &:= W^{(s)}. \end{aligned} \tag{4.17}$$

Assuming that the operator $\mathbf{Prec}_j, j = 1,\ldots,s$, do not change significantly over one Runge-Kutta iteration and that the construction of $\mathbf{Prec}_j$ together with the computation of the block LU-decomposition of th $\mathbf{Tri}_{G_i}$ is a time consuming approach, this simple frozen Runge-Kutta iteration may yield an efficient solution.

4.5 LU-SGS scheme

Considering the algorithm (4.4) as a **smoother** and $\mathbf{Prec}_j^{\text{scalar}}$ is constructed as described in the Section 4.2. Then for each stage in algorithm, we need to solve the linear system

$$\text{Prec}_j^{\text{scalar}} h_j = \alpha_{j+1,j} R(W^{(j-1)}), \tag{4.18}$$

rewriting our preconditioner as,

$$\left((\Delta T)^{-1} M + \alpha_{jj} \frac{\partial R^{\text{scalar}}}{\partial W}(W^{(j-1)})\right) = (L+D+U),$$

where L denotes the lower, D denotes the diagonal, and U denotes the upper part of the matrix. To approximately solve the above equation efficiently, we apply one symmetric Gauss-Seidel sweep. Because of one symmetric Gauss-Seidel sweep our equation can be rewritten as,

$$(D+L)D^{-1}(D+U)h_j = \alpha_{j+1,j} R(W^{(j-1)}). \tag{4.19}$$

And choosing only one stage in the algorithm (4.4), the resulting scheme is known as LU-SGS scheme.

4.6 Explicit Runge-Kutta accelerated by local time stepping

Considering the algorithm (4.4) as a **smoother** and $\mathbf{Prec}_j$ is constructed as described in the section 4.2 with $\alpha_{jj} = 0$. Then for each stage in algorithm, we need to solve the linear system,

$$h_j = \Delta T M^{-1} R(W^{(j-1)}).$$

Approximately solving the linear system by the application of block Jacobi method and choosing the proper stopping criteria, we get exactly a solution algorithm denoted as explicit Runge-Kutta accelerated by local time stepping and the algorithm

can be written as follows,

$$\begin{aligned}
W_i^{(0)} &:= W^n, \\
W_i^{(1)} &:= W^{(0)} - \mathrm{CFL}_{\exp}\alpha_{21}\frac{\Delta t_i}{M_i}R_i(W^{(0)}), \\
&\vdots \\
W_i^{(s)} &:= W^{(0)} - \mathrm{CFL}_{\exp}\alpha_{s+1,s}\frac{\Delta t_i}{M_i}R_i(W^{(s-1)}), \\
W_i^{n+1} &:= W^{(s)}.
\end{aligned} \tag{4.20}$$

Chapter 5

Numerical Examples

In the following section we will show examples with different test cases, in which the solution methods that we discussed in the previous section are used. We would like to test our algorithm for a subsonic and a transonic flow. The test cases are as follows,

(a) Inviscid flow over the NACA 0012 airfoil at the inflow Mach number, $Ma = 0.3$ and angle of attack of 2.0° using the solution algorithms *Implicit scheme based on a derivative corresponding to a discretization of compact stencil*, *LU-SGS scheme* and *Explicit Runge-Kutta accelerated by local time stepping* on coarse grid mesh (160×32).

(b) Inviscid flow over the NACA 0012 airfoil at the inflow Mach number, $Ma = 0.3$ and angle of attack of 2.0° using the solution algorithms *Implicit scheme based on a derivative corresponding to a discretization of compact stencil*, *LU-SGS scheme* and *Explicit Runge-Kutta accelerated by local time stepping* on medium grid mesh (320×64).

(c) Inviscid flow over the NACA 0012 airfoil at the inflow Mach number, $Ma = 0.3$ and angle of attack of 2.0° using the solution algorithms *Implicit scheme based on a derivative corresponding to a discretization of compact stencil*, *LU-SGS scheme* and *Explicit Runge-Kutta accelerated by local time stepping* on fine grid mesh (640×128).

(d) Inviscid flow over the NACA 0012 airfoil at the inflow Mach number, $Ma = 0.8$ and angle of attack of 1.25° using the solution algorithms *Implicit scheme based on a derivative corresponding to a discretization of compact stencil* and *LU-SGS scheme* on coarse grid mesh (160×32).

(e) Inviscid flow over the NACA 0012 airfoil at the inflow Mach number, $Ma = 0.8$ and angle of attack of 1.25° using the solution algorithm *Implicit scheme*

based on a derivative corresponding to a discretization of compact stencil on medium grid mesh (320×64).

In our computations we have chosen the Butcher scheme as,

$$\begin{array}{c|ccc} 0 & 1 & 0 & 0 \\ 0 & \frac{3}{20} & 1 & 0 \\ 0 & 0 & \frac{2}{5} & 1 \\ \hline & 0 & 0 & 1 \end{array}$$

Table 5.1: Butcher Scheme for all computations

We are going to use the CFL ramping strategy to choose the CFL number, which is given as

$$\mathrm{CFL}(n) = \min\{\mathrm{CFL}_{\mathrm{init}} \,.\, f(n),\, \mathrm{CFL}_{\mathrm{max}}\} \tag{5.1a}$$

$$f(n) = \begin{cases} 1, & n < 10, \\ \gamma^{n-10}, & n \geq 10 \end{cases} \tag{5.1b}$$

The parameters $\mathrm{CFL}_{\mathrm{init}}$, $\mathrm{CFL}_{\mathrm{max}}$, γ are given in the description of the examples. The residuals are computed by a volume weighted norm, normalized with respect to the evaluated with free stream values, which is named as density residual and is given by,

$$\text{density residual}(n) := \sqrt{\sum_{j=1}^{N} \frac{(R_{j,\rho}(W^n))^2}{\mathrm{vol}(\Omega_j)^2}} \Bigg/ \sqrt{\sum_{j=1}^{N} \frac{(R_{j,\rho}(W_\infty))^2}{\mathrm{vol}(\Omega_j)^2}} \tag{5.2}$$

All the computations are stopped when the density residual has dropped 12 orders of magnitude or number of iteration has reached 20000, and this is the stopping criteria for all the test cases. We have used 5 symmetric sweeps in the Gauss-Seidal method and preconditioner is not frozen on the multi stage Runge-Kutta scheme.

In order to investigate the examples we additionally compute the well established aerodynamic scalar values and distributions, which is referred from [2]. The scalar values are

1. drag coefficient C_D

2. lift coefficient C_L

The drag and lift coefficients are defined as follows

$$C_D(W) := \frac{2}{\rho_\infty u_\infty^2}\left\langle \int_{\partial\Omega} \langle f_c(W),\, n\rangle\, ds(y),\, g(\alpha) \right\rangle \tag{5.3}$$

and

$$C_L(W) := \frac{2}{\rho_\infty u_\infty^2}\left\langle \int_{\partial\Omega} \langle f_c(W),\, n\rangle\, ds(y),\, h(\alpha) \right\rangle \tag{5.4}$$

where, $g(\alpha) := (0, \cos\alpha, \sin\alpha, 0)^T$ and $h(\alpha) := (0, -\sin\alpha, \cos\alpha, 0)^T$ in which α is the angle of attack.

For an inviscid flow we can calculate the C_D and C_L only by the contribution of forces corresponding to the pressure. Similarly we can calculate the surface pressure distribution C_p also, which is defined as

$$p(W(x)) := \left\langle \langle f_c(W),\, n\rangle\, ds(y),\, (0, n(x), 0)^T \right\rangle, \quad x \in \partial\Omega, \tag{5.5}$$

$$C_p(W(x)) := \frac{2}{\rho_\infty u_\infty^2}(p(W(x)) - p_\infty), \quad x \in \partial\Omega, \tag{5.6}$$

5.1 Test case (a)

We consider an inviscid flow over the NACA 0012 airfoil at the inflow Mach number, $Ma = 0.3$ and angle of attack of 2.0°. We perform the computations using all the three solution algorithms on the mesh of dimension 160×32.

Let us assume $\text{CFL}_{\text{init}} = 1$, $\text{CFL}_{\text{max}} = 1000$ and $\gamma = 1.2$. These numbers are used only for Implicit and LU-SGS scheme. For Explicit scheme we choose $\text{CFL}_{\text{exp}} = 0.75$. The comparison of all the three schemes based on the convergence history of the residuals are shown in the Fig. 5.1.

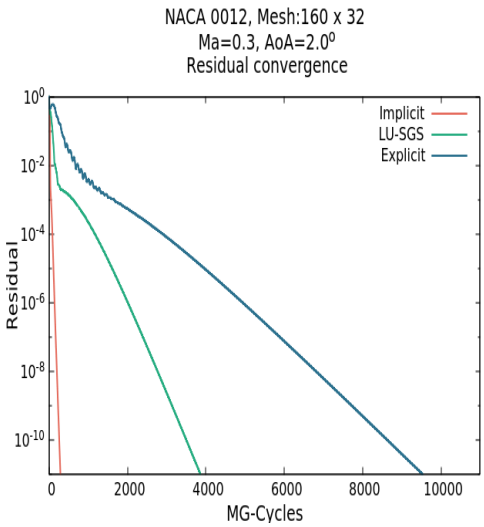


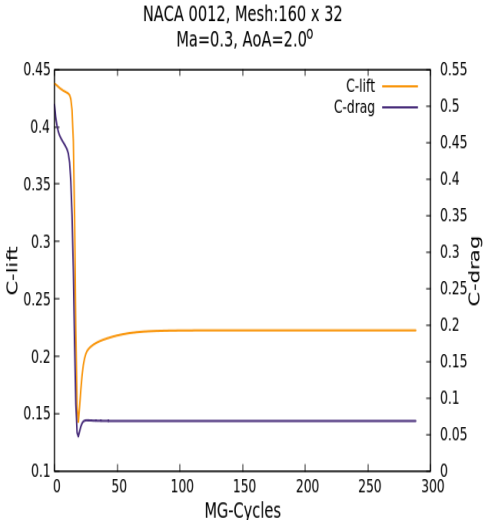
Figure 5.1: Residual convergence, Mesh: 160 $\times$ 32, Mach = 0.3, Angle of attack = 2.0°

From the graph it is observed that the residual is converged very fast by using the implicit scheme after that LU-SGS scheme converges and the explicit scheme is very slower than the other two schemes. It is evident by seeing the CPU time. For the explicit scheme we can be able to note that there is some initial oscillations in the graph. The detailed information of the iteration count and the CPU time are given in the Table 5.2.

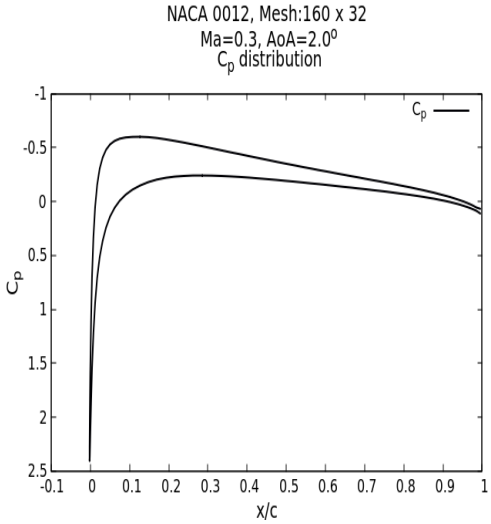
Algorithm	Implicit	LU-SGS	Explicit
No. of iterations	288	3877	9574
CPU time in seconds	73.71	186.542	758.952

Table 5.2: Iteration, Mesh: 160 $\times$ 32, Mach = 0.3, Angle of attack = 2.0°

The computed drag and lift coefficients and surface pressure distribution are shown in the Figs. 5.2-5.4. As expected the drag and lift coefficients and C_p distribution are same in all the algorithms except the initial oscillations of lift and drag coefficients in explicit scheme. From the C_p distribution graph, it is noted that there is loss of accuracy and the trailing edge is not connected properly. That is because of the mesh density, we are using only the coarse grid (160 $\times$ 32).

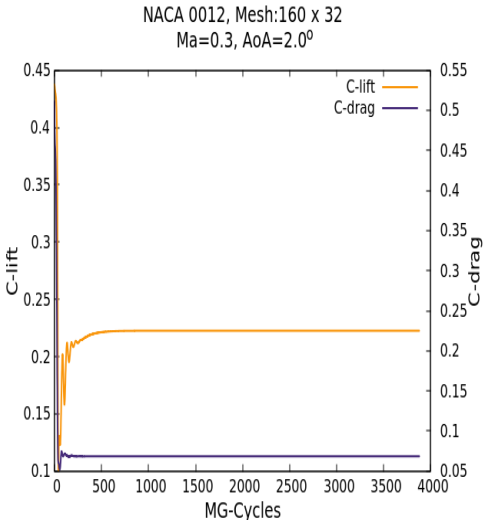


(a) computed lift and drag coefficients

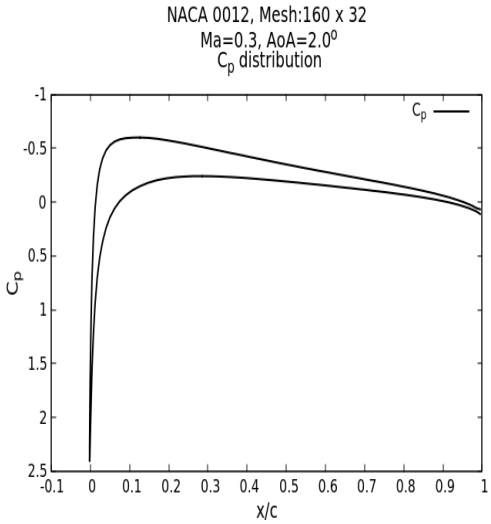


(b) computed surface pressure distribution

Figure 5.2: Implicit scheme, Mesh: 160 × 32, Mach = 0.3, Angle of attack = 2.0°

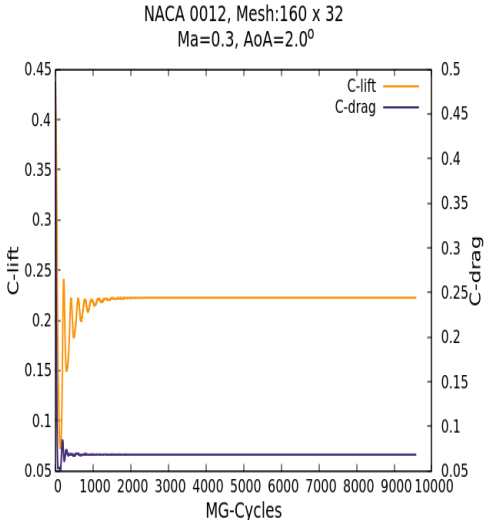


(a) computed lift and drag coefficients

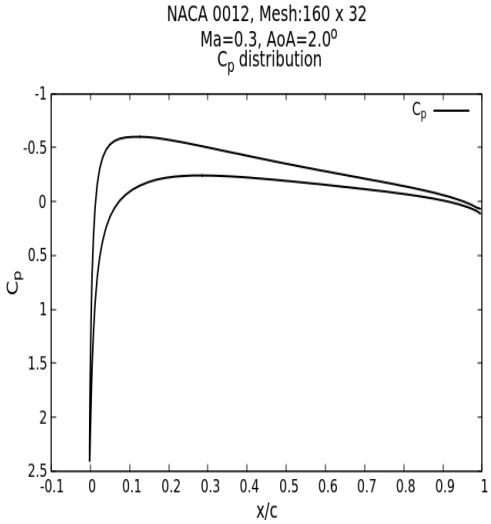


(b) computed surface pressure distribution

Figure 5.3: LU-SGS scheme, Mesh: 160 × 32, Mach = 0.3, Angle of attack = 2.0°



(a) computed lift and drag coefficients



(b) computed surface pressure distribution

Figure 5.4: Explicit scheme, Mesh: 160 × 32, Mach = 0.3, Angle of attack = 2.0°

5.2 Test case (b)

We consider an inviscid flow over the NACA 0012 airfoil at the inflow Mach number, $Ma = 0.3$ and angle of attack of 2.0°. We perform the computations using all the three solution algorithms on the mesh of dimension 320 × 64.

Let us assume the same CFL numbers as the previous test case. The comparison of convergence history of the residuals for this test case are shown in the Fig. 5.5. Also from the CPU time it is seen that the implicit scheme converges faster than the other two schemes in the medium grid mesh also. And for the explicit scheme the oscillations are carried throughout the convergence but using the coarse grid it was only at the beginning. The detailed information of the iteration count and CPU time for the medium grid mesh are given in the Table 5.3.

Algorithm	Implicit	LU-SGS	Explicit
No. of iterations	512	7203	19803
CPU time in seconds	518.54	1274.72	6256.55

Table 5.3: Iteration, Mesh: 320 × 64, Mach = 0.3, Angle of attack = 2.0°

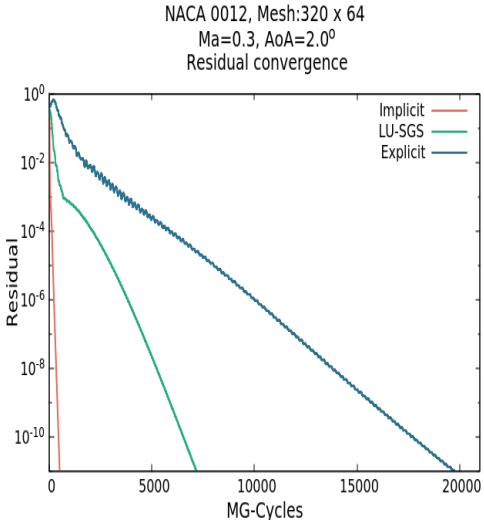
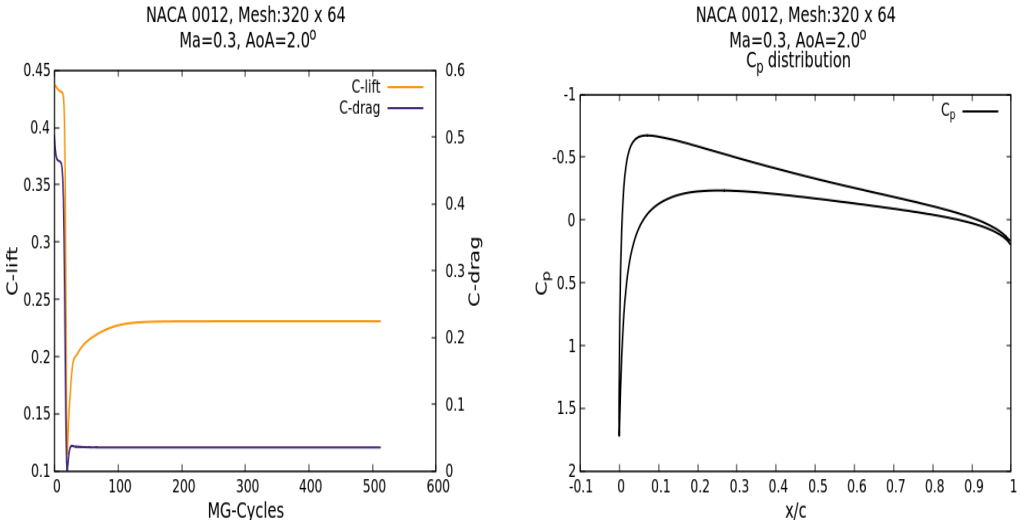


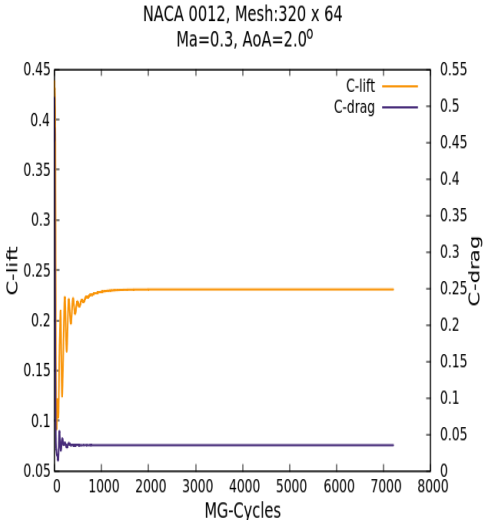
Figure 5.5: Residual convergence, Mesh: 320 $\times$ 64, Mach = 0.3, Angle of attack = 2.0°

The computed drag and lift coefficients and surface pressure distribution for the medium grid mesh are shown in the Figs. 5.6-5.8. From the graph it is noted that the lift coefficient remains almost same as comparing with the coarse grid mesh but the drag coefficients are reduced. So it would give the better performance than the coarse grid. From the distribution graph we can see the increase in the accuracy of the airfoil especially at the trailing edge.

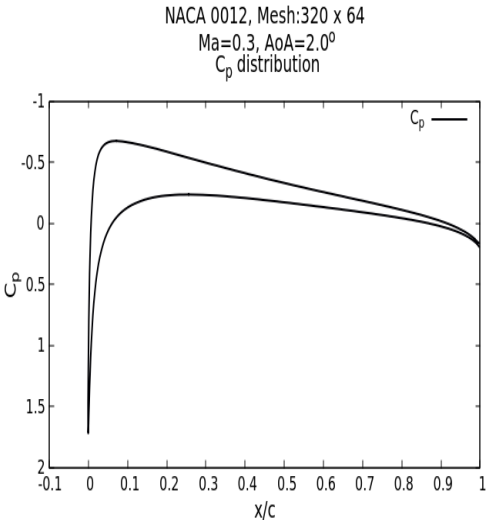


(a) computed lift and drag coefficients (b) computed surface pressure distribution

Figure 5.6: Implicit scheme, Mesh: 320 $\times$ 64, Mach = 0.3, Angle of attack = 2.0°

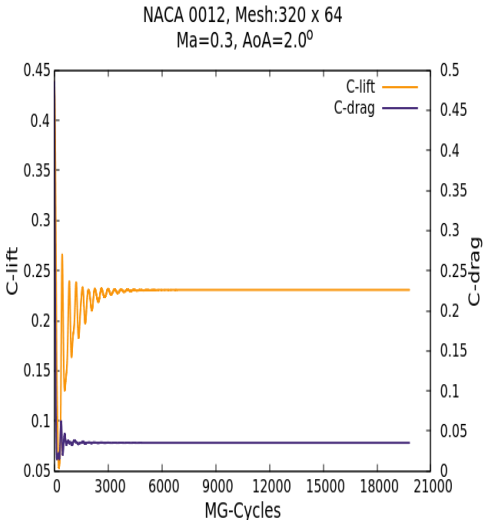


(a) computed lift and drag coefficients

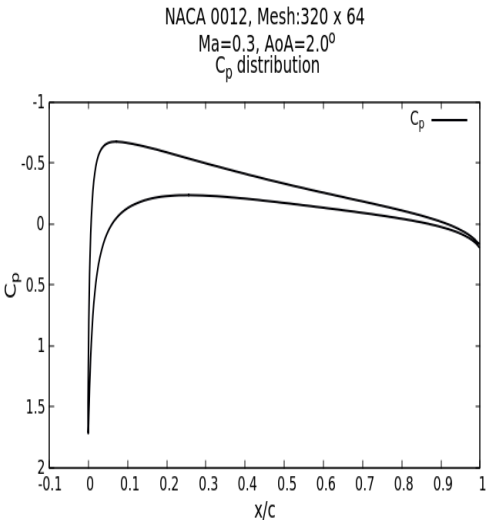


(b) computed surface pressure distribution

Figure 5.7: LU-SGS scheme, Mesh: 320 × 64, Mach = 0.3, Angle of attack = 2.0°



(a) computed lift and drag coefficients



(b) computed surface pressure distribution

Figure 5.8: Explicit scheme, Mesh: 320 × 64, Mach = 0.3, Angle of attack = 2.0°

5.3 Test case (c)

We consider an inviscid flow over the NACA 0012 airfoil at the inflow Mach number, $Ma = 0.3$ and angle of attack of 2.0°. We perform the computations using all the three solution algorithms on the mesh of dimension 640×128.

Let us assume the same CFL numbers. The comparison of convergence history of the residuals for this test case are shown in the Fig. 5.9. Here also the implicit scheme converges faster, LU-SGS scheme converges slower but the explicit scheme never converges and the oscillations are very high. So we stopped whenever the iteration reaches 20000. Being a straight forward method explicit scheme is taking too much of time to converge when we are using fine grid mesh. It is noted that implicit scheme is working nice and it is taking very less number of iterations and the CPU time is also very less. The detailed information of the iteration count and CPU time for the fine grid mesh are given in the Table 5.4.

Algorithm	Implicit	LU-SGS	Explicit
No. of iterations	957	13937	NA
CPU time in seconds	3910.30	9915.93	NA

Table 5.4: Iteration, Mesh: 640×128, Mach = 0.3, Angle of attack = 2.0°

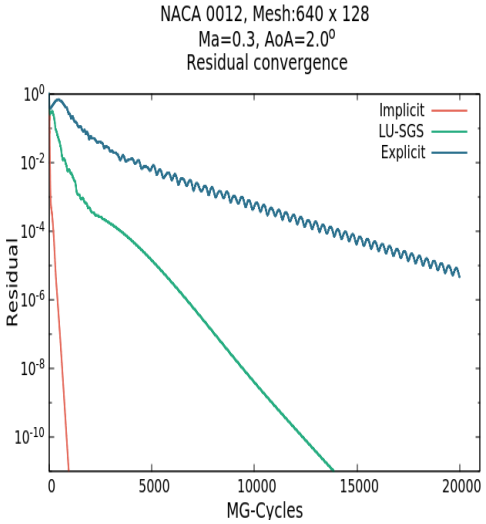
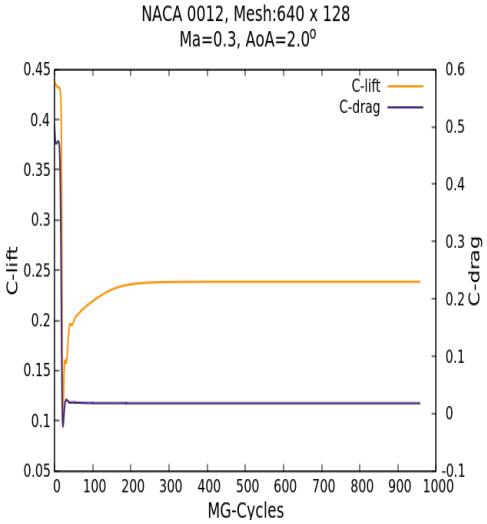
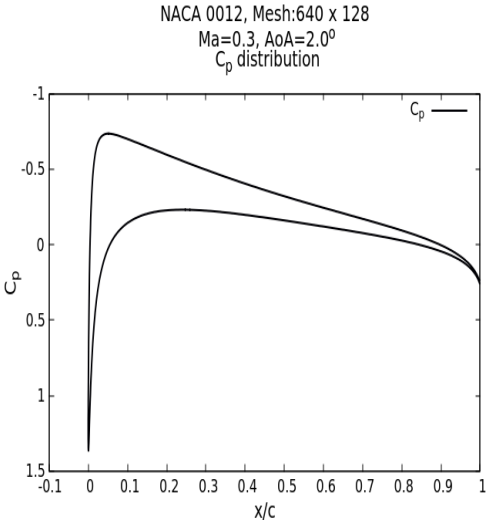


Figure 5.9: Residual convergence, Mesh: 640×128, Mach = 0.3, Angle of attack = 2.0°

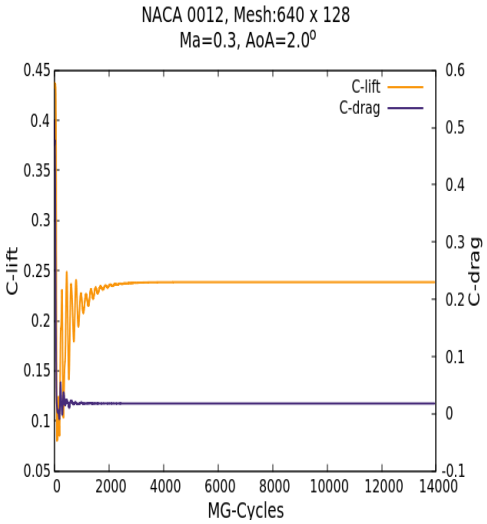


(a) computed lift and drag coefficients

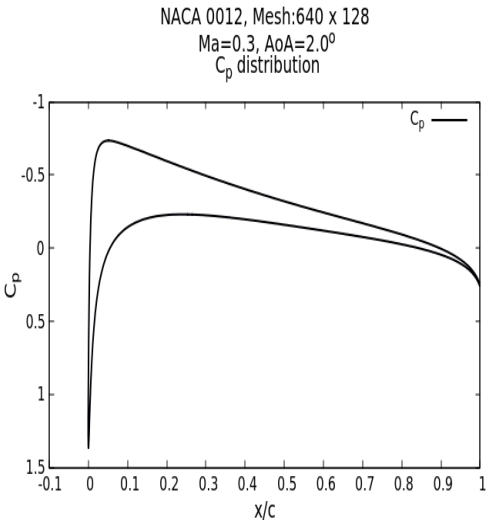


(b) computed surface pressure distribution

Figure 5.10: Implicit scheme, Mesh: 640 $\times$ 128, Mach = 0.3, Angle of attack = 2.0°

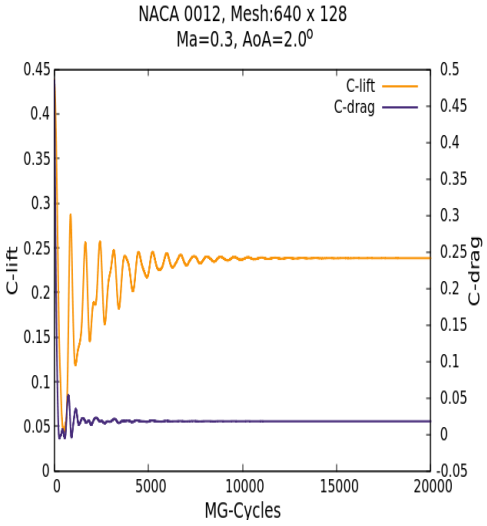


(a) computed lift and drag coefficients

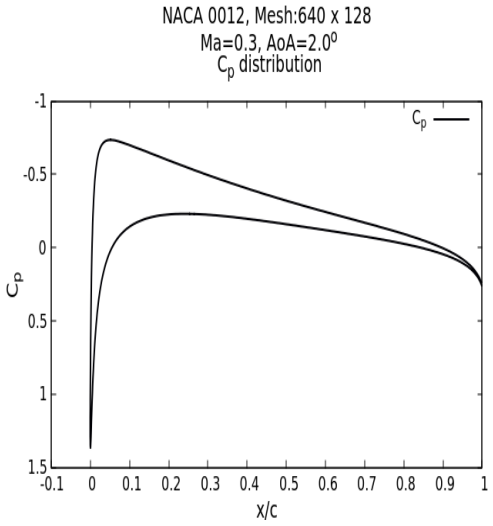


(b) computed surface pressure distribution

Figure 5.11: LU-SGS scheme, Mesh: 640 $\times$ 128, Mach = 0.3, Angle of attack = 2.0°



(a) computed lift and drag coefficients



(b) computed surface pressure distribution

Figure 5.12: Explicit scheme, Mesh: 640 $\times$ 128, Mach = 0.3, Angle of attack = 2.0°

The computed drag and lift coefficients and surface pressure distribution are shown in the Figs. 5.10-5.12. Using the fine grid mesh also we are having the same lift coefficient but the drag coefficient is reduced further. Obviously the performance is also increased with fine grid mesh. In the distribution graph also the accuracy has increased and it is very much noted at the trailing edge of the airfoil.

5.4 Test case (d)

We consider an inviscid flow over the NACA 0012 airfoil at the inflow Mach number, $Ma = 0.8$ and angle of attack of 1.25°. Here we are going to perform the computations only for the implicit and LU-SGS schemes on the mesh of dimension 160 $\times$ 32. It is noted that we could not perform the computations for the explicit scheme in this transient flow condition.

Let us assume $\text{CFL}_{\text{init}} = 50$, $\text{CFL}_{\text{max}} = 1000$ and $\gamma = 1.2$. In order to run our code for this test case we have to alter the CFL number initially. The comparison of these schemes based on the convergence history of the residuals are shown in the Fig. 5.13. From the graph it is observed that both the schemes are converging but the implicit scheme is quiet fast and stable compared to the LU-SGS scheme. It is also evident by seeing the CPU time. The detailed information of the iteration count and CPU time are given in the Table 5.5.

Algorithm	Implicit	LU-SGS
No. of iterations	441	7332
CPU time in seconds	108.71	347.18

Table 5.5: Iteration, Mesh: 160 $\times$ 32, Mach = 0.8, Angle of attack = 1.25°

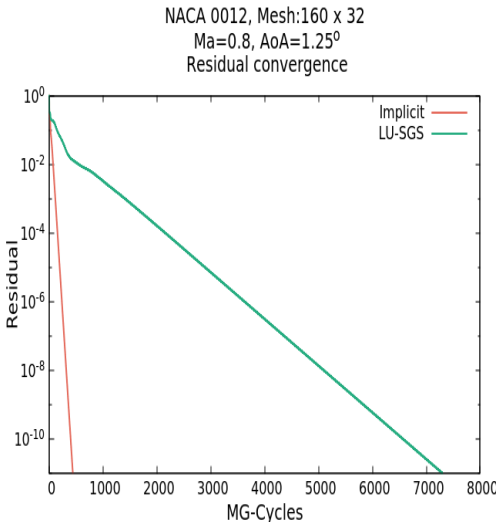


Figure 5.13: Residual convergence, Mesh: 160 $\times$ 32, Mach = 0.8, Angle of attack = 1.25°

The computed drag and lift coefficients are shown in the Figs. 5.14 and 5.15. Both the schemes are giving the same results in drag and lift coefficients.

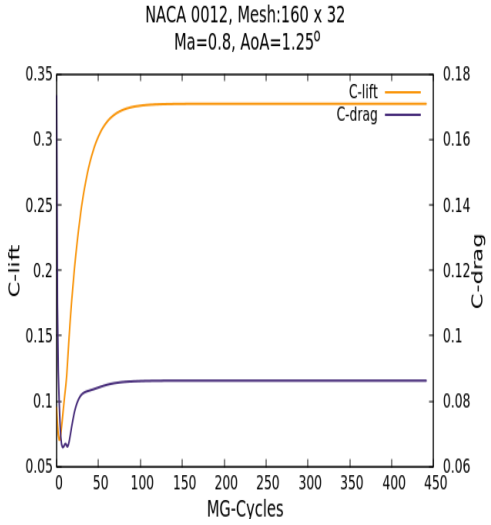


Figure 5.14: Implicit scheme, Mesh: 160 $\times$ 32, Mach = 0.8, Angle of attack = 1.25°

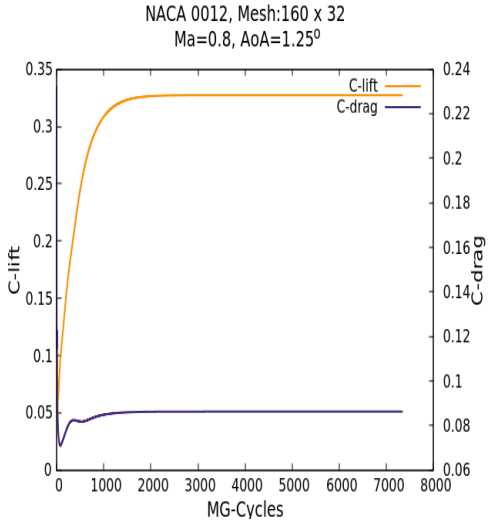


Figure 5.15: LU-SGS scheme, Mesh: 160 $\times$ 32, Mach = 0.8, Angle of attack = 1.25°

5.5 Test case (e)

We consider an inviscid flow over the NACA 0012 airfoil at the inflow Mach number, $Ma = 0.8$ and angle of attack of 1.25°. Here we are going to perform the

computations only for the implicit scheme on the mesh of dimension 320×64.

As discussed in the previous section we have to alter the CFL number initially, which is assumed as $CFL_{init} = 100$, $CFL_{max} = 1000$ and $\gamma = 1.2$. The convergence history of the residual is shown in the Fig. 5.16. It is noted that the implicit scheme is the only method available to run on the given mesh under the given condition. From the graph it is seen that the scheme is converging smoother. The detailed information of the iteration count and CPU time are given in the Table 5.6.

Algorithm	Implicit
No. of iterations	724
CPU time in seconds	718.19

Table 5.6: Iteration, Mesh: 320×64, Mach = 0.8, Angle of attack = 1.25°

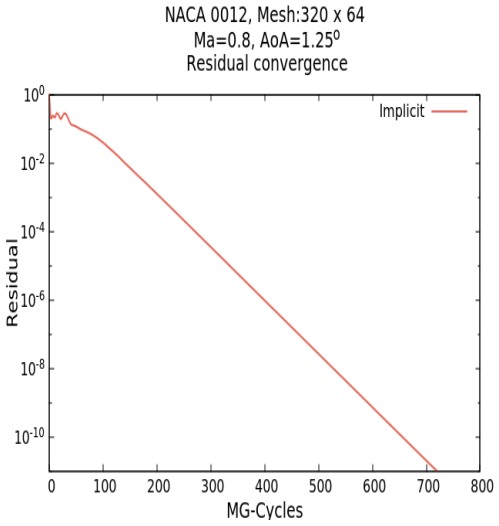


Figure 5.16: Residual convergence, Mesh: 320×64, Mach = 0.8, Angle of attack = 1.25°

The computed drag and lift coefficients are shown in the Fig. 5.17. By comparing it with the coarse grid mesh there exists marginal increase in the lift coefficient and drag coefficient is decreased. This would give us the better performance.

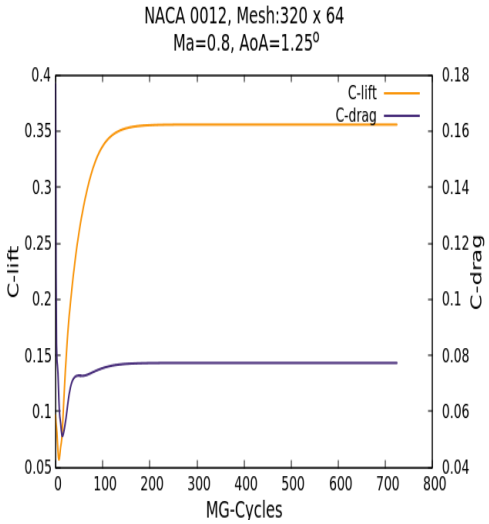


Figure 5.17: Implicit scheme, Mesh: 320×64, Mach = 0.8, Angle of attack = 1.25°

Chapter 6

Conclusion and Future work

From the observation of all the test cases in this thesis, we can find that the implicit scheme is giving the better performance than the other two schemes and it is fast and stable process also. Even LU-SGS scheme is also giving us the desired result but it is time consuming process. The derivative of the residual plays major role in it. As discussed in the solution algorithm chapter the derivative of the residual used in the implicit scheme includes the non diagonal elements also, that is the reason for the fast convergence. In LU-SGS scheme the derivative is approximated only by the spectral radius and it is reduced to the single stage and the explicit scheme is straight forward. By increasing the mesh density we obtain more accurate results.

It is also noted that we can be able to perform computations using all the three schemes on all the three meshes if the test cases are having the subsonic flow condition ($Ma = 0.3$). But for the transonic flow condition ($Ma = 0.8$) we could not perform any calculation using the explicit scheme on our meshes, LU-SGS scheme on medium and fine grid meshes and implicit scheme on fine grid mesh.

In the future we can extend our work by adding some additional features to our algorithm that can really take care of these issues. Some of them are we can replace our first order approximation by the second order approximation in the approximation of the derivative of our numerical flux. We can make use of multi grid algorithm and we can also modify our iterative solution method, which is (Block) Gauss-Seidel, in which we can exploit to line information. It is always interesting to see the changes by altering something in the existing base system.

Bibliography

[1] S. Langer, *Algorithmen zur Lösung der Euler und Navier-Stokes Gleichungen*. 2017.

[2] S. Langer, *Preconditioned Newton methods to approximate solutions of the Reynolds averaged Navier-Stokes equations*.

[3] S. Langer, "Investigation and application of point implicit runge-kutta methods to inviscid flow problems," *International Journal for Numerical Methods in Fluids*, 2011.

[4] S. Langer, "Aglomeration multigrid methods with implicit runge-kutta smoothers applied to aerodynamic simulations on unstructured grids," *Journal of Computational Physics*, 2014.

[5] S. Langer, "Investigation and comparison of implicit smoothers applied in agglomeration multigrid," *AIAA JOURNAL*, 2015.

[6] S. Langer, "Implicit methods and globalization strategies for the robust approximation of solution to the reynolds averaged navier-stokes equations,"

[7] S. Langer, "Hierarchy of preconditioning techniques for the olution of the navier-stokes equations discretized by 2nd order unstructured finite volume methods,"